Procedia Computer Science

Volume 51, 2015, Pages 562–571

ICCS 2015 International Conference On Computational Science

A Model Driven Approach to Water Resource Analysis based on Formal Methods and Model Transformation

Flora Amato[1], Francesco De Paola[2], Crescenzo Diomaiuta[3], Maurizio Giugni[4], Nicola Mazzocca[5], and Francesco Moscato[6]*

[1] DIETI, University of Naples Federico II, Naples(NA), Italy
flora.amato@unina.it
[2] DICEA, University of Naples Federico II, Naples(NA), Italy
depaola@unina.it
[3] DIETI, University of Naples Federico II, Naples(NA), Italy
crescenzo.diomaiuta@gmail.com
[4] DICEA, University of Naples Federico II, Naples(NA), Italy
maurizio.giugni@unina.it
[5] DIETI, University of Naples Federico II, Naples(NA), Italy
nicola.mazzocca@unina.it
[6] DiSciPol, Second University of Naples, Caserta(CE), Italy
francesco.moscato@unina2.it

Abstract

Several frameworks have been proposed in literature in order to cope with critical infrastructure modelling issues, and almost all rely on simulation techniques. Anyway simulation is not enough for critical systems, where any problem may lead to consistent loss in money and even human lives. Formal methods are widely used in order to enact exhaustive analyses of these systems, but their complexity grows with system dimension and heterogeneity. In addition, experts in application domains could not be familiar with formal modelling techniques. A way to manage complexity of analysis is the use of Model Based Transformation techniques: analysts can express their models in the way they use to do and automatic algorithms translate original models into analysable ones, reducing analysis complexity in a completely transparent way.

In this work we describe an automatic transformation algorithm generating hybrid automata for the analysis of a natural water supply system. We use real system located in the South of Italy as case study.

Keywords: Formal Modelling, Model Based, Water Resource Modelling, Hybrid Automata

*Contact Author

 Selection and peer-review under responsibility of the Scientific Programme Committee of ICCS 2015

doi:10.1016/j.procs.2015.05.323

1 Introduction and Related Works

Safety-critical systems are systems whose failure or malfunctioning may have *catastrophic* results like serious injury or even death to people, loss or severe damage of things, environmental harm etc. Risks of this sort are usually managed by using methods and tools of safety engineering where typical design methods include probabilistic risk assessment, failure analyses etc. These systems are increasingly computer-based and techniques for their analysis and forecasting usually belong to the field of System and Software Engineering. For long time in literature this term was coupled to transport systems, video-surveillance, space flights, banking system etc.

Anyway, practically every *natural* system is Safety-Critical even if no software-based control system is applied: earthquakes, epidemics, and natural water supply systems as well. In the last case, life-criticisms are both in excesses and in lacks of water supply for humans in a region. Lacks lead to dryness and excesses to floods. Every year, thousands of lives and property worth billions of dollars are lost in floods all over the world [1]. Research Literature proposes several approaches to critical infrastructures and systems modelling issues. Almost all, especially in *natural* or *social* systems management, rely on simulation techniques (for water management you can see, for example: [2, 3, 4]). Simulation enables forecasting, and helps to evaluate the effects of fault and unpredicted events. Such kind of qualitative analysis is known as "what if" study and it is able to demonstrate the possibility that a certain undesired event can happen, but is not able to demonstrate that a certain undesired event can *never* happen. Generally speaking, simulation does not allow to analytically evaluate quantitative dependability attributes and to verify properties on the model, like Safety related ones. In order to overcome such limitations, formal methods are widely used in the community of dependability experts in order to evaluate the attributes of interest. The main problems in the formal analysis of *natural systems* are: (a) the systems to model are extremely complex and heterogeneous: they are both dynamic and event-based; (b) Experts in *natural* systems are usually not computer scientists and the models they use to describe systems are not easily analysable in terms of safety and dependability.

In this work we focus on natural water supply systems, but the methodology is applicable to many other types of systems. Heterogeneity in the systems we are going to explore is in the fact that they have important temporal aspects and potentially involve both continuous variables and discrete events. This characteristics motivates the use of hybrid systems modelling. In the scientific literature, the use of formal methods in system engineering and analysis is common in the software domain. In particular, Model Driven Engineering (MDE) and Model Based Transformation [5] methodologies are now going to be widely used by enterprises too in order to prevent erroneous design and the introduction of too much errors during development phases. In the last years, several works have tried to apply formal methods to natural and social systems. In particular for hydrology, probabilistic and statistical methods have been investigated for analyses. For example, Bayesian analysis is applied in [6] and authors of [7] and [8] apply statistical analyses to natural and social systems. Anyway, the approaches presented in the literature do not take into account of dynamics of the system to analyse. At the best of our knowledge, the only work that uses formal methods for the modelling of natural dynamic systems is [9]. Differences with the work presented here are in the way the hybrid automata are generated: we use model transformation reading models that are defined in a (graph-based) language known by most of hydrology experts.

This paper shows an approach for formal modelling of water supply system in order to analyse safety critical events. The approach is based on a technique known as *Model Transformation*. It allows for the modelling of systems with formalisms well known to the experts in hydrology. Classic models are processed in order to generate models analysable with model

checking techniques applied to Hybrid Models (in particular, Hybrid Automata).

The paper is organized as follows: section 2 describes the methodologies based on Model Transformation; section 3 introduces formal modelling and hybrid automata. In section 4 we describe the target scenario. Section 5 contains the description of our modelling methodology and the algorithm we use to enact model transformation. Finally, section 7 reports some concluding remarks.

2 Model Driven Engineering and Model Based Transformations

Model-Driven Engineering (MDE) methods and techniques try to solve the problem of system modelling and deployment by facilitating definition, composition, implementation and verification of complex systems. Usually in MDE methodologies, system design refers to models (high-level description of systems) defined by formal languages; formal rules are enacted in order to transform high level descriptions to low level ones preserving models soundness. This leads to implementation of systems correct by construction where requirements are validated during all life cycle. In MDE, functional and non functional properties have to be verified at early design phase, in order to produce correct implementations through model transformation.

In general, a *model transformation*[10] takes a model as input and generates a model as output. From the point of view of languages and formalisms used to define input and output models, two kinds of model transformations exist: formalisms for input and output models are the same in *endogenous* transformations, different in *exogenous* ones. Endogenous transformations rewrite the input model to produce the output model. They are applied for different tasks such as model refactoring, optimization, evolution, and simulation, to name just a few. Furthermore, transformations can produce models at the same level of abstraction (*horizontal* transformations) or at different levels of abstractions (*vertical* transformations). Exogenous transformations are usually used in conjunction with vertical transformation, building the base for practices like code generation from design models. Horizontal transformations are of specific interest to realize different integration scenarios, e.g., translating a UML class model into an Entity Relationship (ER) model etc. Various model transformation approaches have been proposed in the past years, mostly based on either a mixture of declarative and imperative concepts, such as ATLAS Transformation Language (ATL) and Query/View/Transformations (QVT) [11] or Real Time Agent Modeling Language RT-AML and MetaMORP(h)OSY MDE framework ([12, 13, 14]).

Summarizing, all approaches describe model transformations by rules using metamodel elements, whereas the rules are executed on the model layer for transforming a source model into a target model .

3 Hybrid Systems and Formal Modelling

As introduced before, the system we are going to model are naturally *hybrid*. Models of hybrid systems have to take into account of continuous evolution and discrete mode transitions.

Hybrid modelling paradigms [15, 16, 17], solve problems in exhaustive analyses of models due to continuity by using mechanisms that model discrete state changes.

In Hybrid Models, differential equations form a common representation of continuous system behaviour. The system is described by a state vector. Behaviour over time is specified by composition of differential equations and interaction with the environment is specified by input

and output signals. Partitioning in continuous state space transforms the continue models into discrete one. This leads to a *state machine* which consists of a set of discrete modes whose changes are caused by events and specified by state transition function. A transition may produce additional discrete events, causing further transitions.

Hybrid Automata are one of the most used formalism for the formal description of hybrid models. A hybrid automaton H is defined by the following entities: (a) a finite directed multi-graph $G = (V, E)$. Where vertices V represents systems discrete states; (b)a finite ordered set $X = x_1, x_2, \cdots, x_n$ of real valued; (c) variables for modelling the state variables of continuous components; (d) an initial node and an assignment of initial values to variables in X (e) for each node $v \in V$, a specification of flow conditions for variables in X. This is typically done using differential equations that govern the evolution of the real valued variables when the system is in the mode v; (f) for each node $v \in V$, a specification of invariance requirements on variables in X. These requirements are specified as finite conjunctions of linear inequalities on the real valued variables;

Complex hybrid systems that consist of several communicating hybrid systems executing concurrently can be modeled by composing hybrid automata and using appropriate mechanisms for synchronization and message passing. We call these automata *product automata*.

4 Application Scenario: Water Resources Management

This section describes a simple case study, which refers to the analysis and modelling of the aqueduct of Serino, which is located in an Italian town with more than 7000 inhabitants near Avellino in Campania, famous for its aqueduct built in Roman age.

The source of water is located in Urciuoli, which is about 338 m on the mean sea level (s.l.m.m.), with a flow rate of approximately 1100 l/s. Water coming from the sources Acquaro - Pelosi (approximately 370 m s.l.m.m.) flows into this source with a flow rate of about 900 l/s. These waters then flow into a pool located at a height of 322.5m passing through a steel pipe (DN700) and than it converges in the original channel of the aqueduct of Serino. The output flow feeds several cities in the region and finally arrives into the pools located on the hill of Cancello (a city near Caserta) and the latter then branches off to other Italian towns.

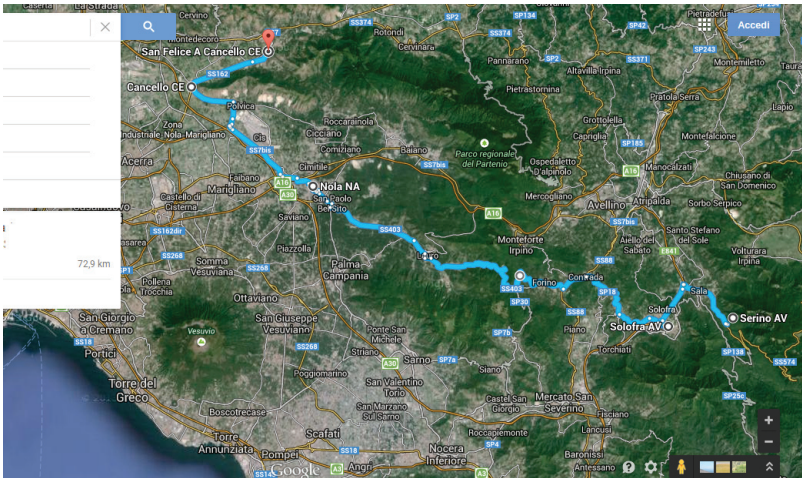

Figure 1: Serino Aqueduct path

Fig.1 shows the approximative path from Serino to Cancello on Google Maps.Fig.2 shows an operating diagram of the aqueduct where we consider only significant elements of the system.

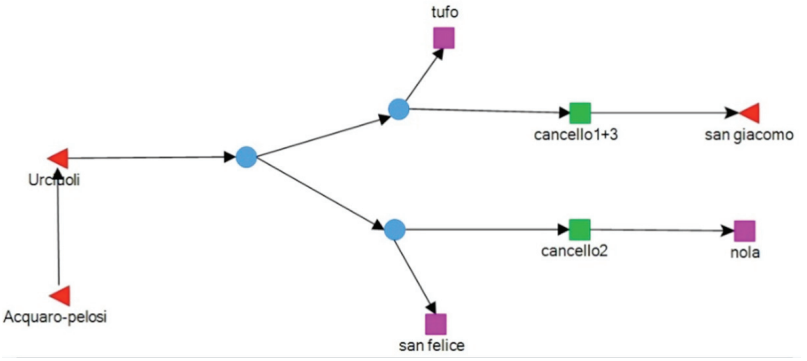

Figure 2: Serino Aqueduct Model

In the figure, triangles represent Reservoirs, purple squares represent demand nodes while the green ones represent sinks. Blue circles depicts non storage nodes, like split point in the flow.

In particular, the reservoirs on the left of Fig.2 models the two water sources of Urciuoli and Aquaro-Pelosi; Non Storage nodes are related to confluences points of water flows. Tufo and San Felice and Nola are demand nodes that request for consumption by some neighbouring towns. The two Network Sinks of Cancello are collection pools on hill Gate. Finally San Giacomo is an outflow plant reservoir.

In order to exploit model transformation for critical analysis, we need a model expressed in a formalisms well known to Hydraulic Engineers. The model will be transparently translated into hybrid automata in order to perform further analyses on the water supply system. The model we use here is a graph model which is usually used for water supply systems simulation [18].

For simplicityś sake, we simplified the graph. Fig.3 contains the reduced version: it consists of the following three groups of elements : (a) the water sources are defined as a single water storage node; (b) the two nodes representative of water demand along the route; (c) the two pools of Cancello;

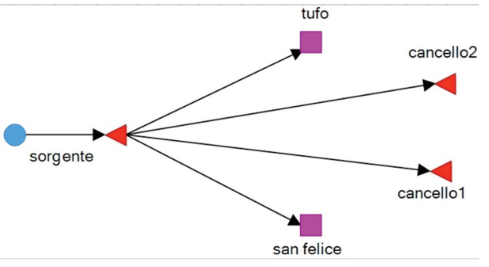

Figure 3: Aquatool Reduced Model

5 Modelling and Model Based Transformation in Reservoirs Management

The steps to enable hybrid automata generation for the whole system modelling are the following: (1) We must provide an hybrid automaton for each basic components of the water resource model; (2) We must analyse the graph(Fig.2 and3) containing the water supply topology and configuration in order to build the hybrid product automaton of the whole system and to instantiate its parameters.

These 2 steps enact the Model Transformation process introduced before. In this model we assume that reservoirs have an input and and output gate. Gates can be managed by valves. Input gates, when input valves are open, fill reservoirs with a given flow of water, while output gates let water to flow out into channels and towards next reservoirs and demand units. Reservoirs can have unmanaged input sources of water (like rain, water from rivers etc.). Their configuration includes the minimum and the maximum threshold of waters they have to manage.

Time to open(t_{open}) and *time to close*(t_{close}) parameters characterizes the delays in opening and closing valves respectively.

Let q_r and $\dot{q}_r$ be the the current water level and its rate of change respectively at a generic reservoir; let p_r and $f_i$be the input water rates due to environmental sources and input source respectively and let f_o be the water rate out coming from the reservoir. In general:

$$\dot{q}_r = p_r + f_i - f_o$$

The Demand nodes are similar but we suppose (for simplicity) that out gates are ever open.

In addition, we suppose all flows constant for simplicity. This is not a limitation since the analyses execute in short time intervals, compared to the dynamics of the real system, and it is possible to re-parametrize models in order to face environmental changes. Fig. 4 shows the hybrid automata modelling Reservoirs (on the left) and Demand nodes (on the right).

A reservoir works into four main states: **Fill**, when the output valve is closed and the input in open; **Stop**, when only unmanaged input flows enter the reservoir; **Fill and Drain** where both input and output valves are open and **Drain** when only output valve is open. In addition,three intermediate states models time passing during valves opening and closing operations (**OpenOut**, **CloseIn** and **CloseOut**) For simplicity we consider the operation of opening input valve instantaneous. During the first four states, the invariant in the node enables the evolution of the continue variable q_r. The *clock* variable t records the time passed in each state (in this example it resets on every state transition). The variable *Request* and the events *ReqIn*, *ReqOut*, *EndReqIn* and *EndReqOut* manage synchronizations among different automata instances. In particular, guards ending with an exclamation mark are events generated by the automaton, those ending with a question mark are generated by other automata and waited by the current one.

The abstract behaviour of the automaton is simple: proper sources fill a reservoir and input gates are blocked when reservoir is full. Output gates open when request from downstream elements (this takes some times). Then, input flows still fill the reservoir while another stream flows out. If the level of the reservoir is too high, input valve is closed and it works in drain only mode. When too few water remains in the reservoir, input gates are opened again. Downstream requests are managed in this way. If the reservoir has the output valve closed and a new request arrives, the valve is open. Every time a Request arrives or ends, a *Requests* variable takes into account the number of active requests. If no requests are active during one of the two draining

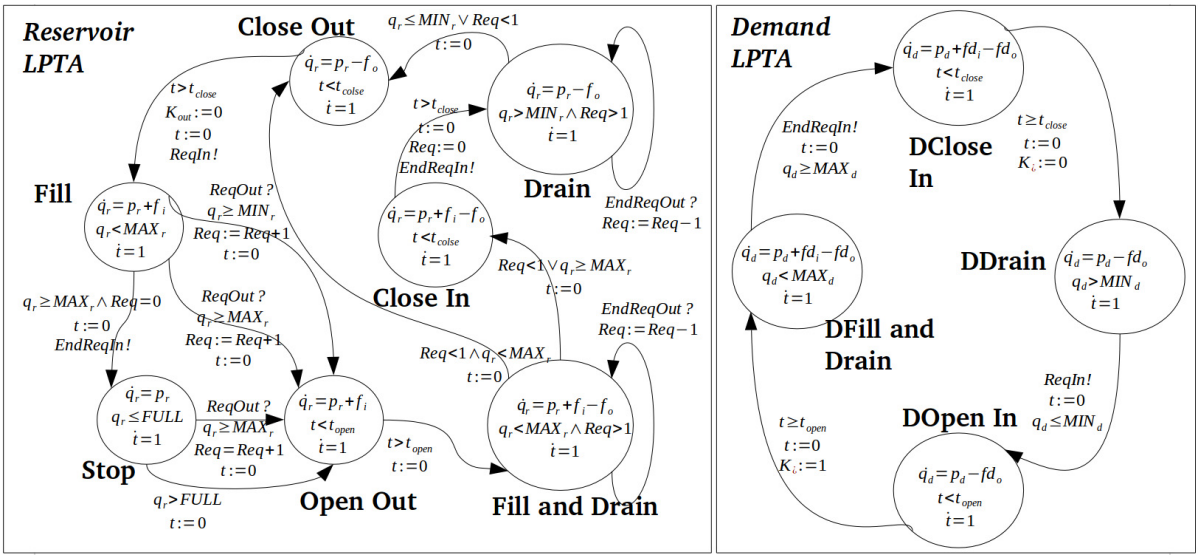

Figure 4: Hybrid Automata of Water Management Elements

phases, The output valve is closed. Demand nodes are similar but simpler and their description is omitted for brevity's sake.

The Model Transformation algorithm from the models in Aquatool format to hybrid automata is reported in the following.

Data: Q a queue of Nodes to analyse; V and E set of Nodes and Edges in the source Graph
Result: $SynReqIn$, $SynReqOut$, Lists of Synchronization for Product Hybrid Automata
$Q = \{v \in V : outgoing(v) = \emptyset\}$;
while Q *is not empty* **do**
  n=Q.dequeue();
  for $sr_{in} \in$ *Set of* Req_{in} *of* n **do**
    $SynReqIn$.add(sr_{in});
  end
  for $sr_{out} \in$ *Set of* Req_{out} *of* n *(if any)* **do**
    for $t \in outgoing(v)$ **do**
      link with the same name sr_{out} to $to(t).Req_{in}$;
      set the input flow parameter of $to(t)$ to: $\alpha * v.q_{out}$
      (where α is the part of flow from v routed to $to(v)$);
    end
    Q.enqueue(all $w \in from(incoming(v))$);
  end
end

Algorithm 1: Model Transformation Algorithm

6 Analyses on Translated Model

In this work we study safety properties of the water plant by applying model checking techniques to the models described in the previous section. Model Checking [19] allows for an exhaustive

analysis of the state space of the model facilitating the discovery of possible dangerous situations. It enables the identification of safety hazards that may remain hidden when using a classic simulation-based study of the System.

In order to verify safety properties on LPTA models:

1. we must instantiate models with known parameters, like the input value of q_r and q_d, flow partitioning in split nodes etc.;
2. we must generate a product automaton for the whole system by applying Algorithm 1;
3. we must describe safety properties by using logic proposition: in this way the model checker will be able to verify the properties on the whole model.
4. we must run a model checker compliant with the model's formalism and the logics used to define properties.

In this work we use UPPAAL Cora[1] in order to enact model checking on hybrid automata. The Algorithm 1 generates an XML configuration for the tool. Notice that UPPAAL Cora instantiates automaton by using a global configuration file. Hence, in order to define the whole model, we need the description of automata to instantiate and compose (we have already discussed automata before and Fig.4 depicts them).

Safety properties in UPPAAL Cora are expressed by Computational Tree Logic (CTL[19]) formula. We do not want to introduce the logic here and CTL syntax, but we want to address simple safety property and report the result of the analysis of this formula by means of the model checker. The Safety property we want to study is:

Will it never happen that No Request arrives from downstream demand nodes and reservoirs and that the source reservoir of Serino overflows?

When the property is checked a deadlock is detected in the system: this is due to the fact that the model does not evolves when the Serino reservoir is FULL and something from the environment (for example, rain) increments the volume of the water in the reservoir, possibly causing dangerous flooding.

7 Conclusions and Future Work

We have described an approach for formal modelling of water supply system in order to analyse safety critical events. The approach is based on a technique known as *Model Transformation*. This technique allows for the modelling of the system with models and languages well known to the experts of hydrology. Classic models are processed in order to generate models analysable with model checking techniques applied to Hybrid Models (in particular, Hybrid Automata). Safety properties can be checked on the target, analysable, models. Presence of deadlocks and fails of formulas checks are symptom of problems in the system model and the presented methodology offers a fast way to analyse system models in order to detect problems that are difficult to trace in simulation.

8 Acknowledgements

This work was funded through the project PON *BE & SAVE-AQUASYSTEM-SIGLOD* - CODE PON04a2.

[1] `http://people.cs.aau.dk/~adavid/cora/introduction.html`

References

[1] Abhas K Jha, Robin Bloch, and Jessica Lamond. *Cities and flooding: a guide to integrated urban flood risk management for the 21st century*. World Bank Publications, 2012.

[2] Jens Grundmann, Niels Schütze, Gerd H Schmitz, and Saif Al-Shaqsi. Towards an integrated arid zone water management using simulation-based optimisation. *Environmental Earth Sciences*, 65(5):1381–1394, 2012.

[3] JG Leskens, M Brugnach, and AY Hoekstra. Application of an interactive water simulation model in urban water management: a case study in amsterdam. *Water Science & Technology*, 70(11):1729–1739, 2014.

[4] Katalyn A Voss, James S Famiglietti, MinHui Lo, Caroline Linage, Matthew Rodell, and Sean C Swenson. Groundwater depletion in the middle east from grace with implications for transboundary water management in the tigris-euphrates-western iran region. *Water resources research*, 49(2):904–914, 2013.

[5] Stuart Kent. Model driven engineering. In *Integrated formal methods*, pages 286–298. Springer, 2002.

[6] Alberto Viglione, Ralf Merz, José Luis Salinas, and Günter Blöschl. Flood frequency hydrology: 3. a bayesian analysis. *Water Resources Research*, 49(2):675–692, 2013.

[7] Congli Dong, Gerrit Schoups, and Nick van de Giesen. Scenario development for water resource planning and management: a review. *Technological Forecasting and Social Change*, 80(4):749–761, 2013.

[8] Flora Amato, Antonino Mazzeo, Vincenzo Moscato, and Antonio Picariello. Exploiting cloud technologies and context information for recommending touristic paths. In *Intelligent Distributed Computing VII*, pages 281–287. Springer, 2014.

[9] MV Panduranga Rao and Akhilesh Chaganti. Safety verification of floodgate operation protocols using hybrid automata. In *Proceedings of the International MultiConference of Engineers and Computer Scientists*, volume 1, 2014.

[10] Tom Mens and Pieter Van Gorp. A taxonomy of model transformation. *Electronic Notes in Theoretical Computer Science*, 152(0):125 – 142, 2006. Proceedings of the International Workshop on Graph and Model Transformation (GraMoT 2005), Graph and Model Transformation 2005.

[11] Marcel Van Amstel, Steven Bosems, Ivan Kurtev, and Luís Ferreira Pires. Performance in model transformations: experiments with atl and qvt. In *Theory and Practice of Model Transformations*, pages 198–212. Springer, 2011.

[12] Rocco Aversa, Beniamino Di Martino, and Francesco Moscato. Critical systems verification in metamorp (h) osy. In *Computer Safety, Reliability, and Security*, pages 119–129. Springer, 2014.

[13] Francesco Moscato, Flora Amato, Alba Amato, and Rocco Aversa. Model–driven engineering of cloud components in metamorp (h) osy. *International Journal of Grid and Utility Computing*, 5(2):107–122, 2014.

[14] Flora Amato, Anna Rita Fasolino, Antonino Mazzeo, Vincenzo Moscato, Antonio Picariello, Sara Romano, and Porfirio Tramontana. Ensuring semantic interoperability for e-health applications. In *Complex, Intelligent and Software Intensive Systems (CISIS), 2011 International Conference on*, pages 315–320. IEEE, 2011.

[15] Rajeev Alur, Costas Courcoubetis, Thomas A Henzinger, and Pei-Hsin Ho. *Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems*. Springer, 1993.

[16] John Guckenheimer and Stewart Johnson. Planar hybrid systems. In *Hybrid systems II*, pages 202–225. Springer, 1995.

[17] Pieter J Mosterman and Gautam Biswas. Formal specifications for hybrid dynamical systems. In *IJCAI (1)*, pages 568–577, 1997.

[18] Joaquín Andreu, Jy Capilla, and Emilio Sanchís. Aquatool, a generalized decision-support system for water-resources planning and operational management. *Journal of hydrology*, 177(3):269–291,

1996.

[19] Edmund M Clarke, Orna Grumberg, and Doron Peled. *Model checking*. MIT press, 1999.